

Original Article

Graph-Augmented AI for Hierarchical Reasoning over Enterprise Codebases and Regulatory Text

Santosh Srinivasaiah¹, M Rajeswari²

¹Electronics and communication engineering, Tadipatri Engineering college.

²Department of Electronics and Telecommunication Engineering, Bangalore Institute of Technology, Bangalore, Karnataka.

¹Corresponding Author : santosh@talerresearch.com

Received: 25 February 2026

Revised: 18 June 2026

Accepted: 24 June 2026

published: 30 September 2026

Abstract - Graph-Augmented AI Systems are on the cusp of a paradigm shift, marking the next significant leap in deep-level reasoning within risk-complex enterprise code and regulatory documents. As part of this activity, this paper proposes an innovative algorithmic result and solution using the concept of Hierarchy Graph Augmented Reasoning Network (HiGAR-Net), an innovative approach goal to aimed at modelling complex-risk semantics through both graph and deep-level reasoning. Through graph augmentation, the proposed approach is able to provide high interpretability and scalability. Experimental results show that HiGAR-Net achieves a high weighted precision of 95%, a weighted recall of 93%, and a weighted F1 of 95% with good class-wise performance in the compliance classification task. Although the proposed framework is intended to be multimodal extensions friendly, the present experimental analysis of the framework concentrates on graph-based representations based on enterprise codebases and regulatory writing.

Keywords - Graph-Augmented AI, Hierarchical reasoning networks, Enterprise code intelligence, Regulatory document analysis, HiGAR-Net, Semantic dependency modeling.

1. Introduction

The code bases of enterprises and regulatory documents are developing and growing to an untold extent, triggered by digitisation and constantly shifting regulations [1]. These data sources are rich or high in interdependencies, semantics, and restrictions belonging to a specific department field that render automaton reasoning highly challenging. Conventional models of deep learning would not be able to capture hierarchical models and relationships defining compliance in heterogeneous sources of data, such as code bases and documents. Including models would face shortcomings in accuracy and reliability [2]. The challenge emerges in making reasoning by computers possible in both technical and regulatory fields of data. Code bases of enterprises' requirements and needs of a semantics understanding to facilitate reliable and efficient execution [3].

To address the demands of the target of the enterprise, the AI system must help to support not only a high degree of accuracy but also a transparent and explainable outcome of data [4]. The research motivation lies in creating a system that removes the gap between technical risk and complexities and regulatory requirements, making it possible for the enterprise to rely on AI for reasoning in mission-critical scenarios [5]. To this end, this research work, as presented, proposes a new approach called the Hierarchical Graph-Augmented

Reasoning Network (HiGAR-Net) system, which integrates hierarchical graph augmentation and neural reasoning. The proposed system models semantic dependency for multiple layers of code in the enterprise, as well as regulatory texts, to help and support a deeper reasoning system that remains scalable as well as transparent [6].

The proposed HiGAR-Net system clearly achieves the goal of the new benchmark for AI reasoning in the enterprise, as it reaches an accuracy of 99% in experimentation. The research work remains a milestone for future extensions, including multimodal reasoning, ethical AI, and real-time deployment for mission-critical AI reasoning in the enterprise [7]. The proposed system clearly serves as a cornerstone for AI graph augmentation, which removes the gap existing between technical reasoning and regulatory requirements for a transparent and accountable system [8]. Contemporary enterprises are characterized by massive graphs of code dependencies, legacy code, and ever-changing regulations. These environments pose high-performance reasoning problems that include processing dependencies in code as well as in regulations over various geographies.

However, changing regulations, which are updated on a frequent basis with intricate interpretations, pose great challenges [9]. Without advanced reasoning capabilities,



enterprises are at the highest risk and complexity of errors, attacks, or violations of compliance regulations. Several deep learning-based architectures, including transformers and Graph Attention Networks, perform excellently in standalone complex tasks like code summarisation and document classification. However, they are weak in providing hierarchical reasoning capability, which cumulatively enables code reasoning from the perspective of the entire enterprise as well as analysing compliance regulations as documents. Hierarchical Graph Augmentation offers robustness [10]. This provides precise reasoning capabilities combined with enhanced explanation. Compliance-based industries demand a high level of explanation in artificial intelligence systems to gain the highest trustworthiness [11]. Hence, this area emphasises the requirement of providing higher-level transparency in AI designs.

The proposed architecture, Hierarchical Graph-Augmented Reasoning Network (HiGAR-Net), overcomes several limitations by providing enhanced code syntactic dependencies, semantic dependencies, and compliance dependencies in a unified code reasoning model [12]. It enables state-of-the-art code reasoning capabilities combined with the highest transparency in compliance regulations. Therefore, it sets a benchmark in improving the development of code reasoning-based AI industries. HiGAR-Net has several implementations in the future in terms of multimodal reasoning, real-time processing, and explanation platforms in AI development.

Therefore, it has several implementations in the near future in terms of providing enhanced reasoning capabilities in various AI data industries [13]. Enterprise code bases and compliant documents are themselves complex and risk structures, possibly entailing nested dependencies, a domain for target semantic reasoning, and compliant reasoning [2]. Conventional models fail to model complexities and risk owing to their linear or flat, full structures. Moreover, the dynamic nature of business rules and evolving regulations adds fuel to the fire, rendering static reasoning mechanisms ineffective in decision-making [14].

Though effective for sequential attention modelling, transformer architectures possess no intrinsic relation inductive bias and although relation-aware, Graph Attention Networks are not inherently hierarchical, lacking any form of inductive bias across enterprise and regulatory layers [15]. HiGAR-Net is based on the core concepts of graph representation learning [16], where nodes represent entities and edges represent syntactic, semantic and compliance relationships between them in a single learnable structure. In addition to being technically novel, HiGAR-Net helps enterprise code intelligence by extending graph-based program representation paradigms [17] to compliance-sensitive reasoning. HiGAR-Net combines the neural message passing with hierarchical differentiable pooling [18] to

support weighted compliance attention and semantic alignment in one end-to-end network. It features a graph-based reasoning that facilitates explainable decision processes that align with the principles of interpretable and responsible AI [19] and meet regulatory and ethical standards.

Compared to the current GraphRAG, GNN+LLM, and hierarchical GNN methodologies, the offered HiGAR-Net incorporates the principle of compliance in the hierarchical graph reasoning framework, where regulatory constraints are implemented directly in the message transfer and attention schemes, but not implemented as the post-hoc retrieval or rule validation. The model collaboratively carries out semantic reasoning, hierarchical dependency modelling and compliance alignment on a based unified graph structure, allowing traceable and regulation-consistent inference. This architectural combination makes HiGAR-Net different to the previous retrieval-augmented and language-based reasoning systems.

2. Related Work

2.1. Graph Neural Network in Enterprise AI

Transformer	Graph Attention Network	Hybrid Approach
Contextual understanding	Strong	Strong
Semantic Dependence Modeling	Limited	Robust
Scalability	Moderate	Moderate
HiGAR-Net		

Fig. 1 Graph neural network in enterprise AI

The above Figure 1 represents a Graph Neural Networks (GNNs) that have emerged as a fundamental paradigm within enterprise AI for their ability to extract and encode relational entities from complex and risk datasets. Through the encoding of interdependencies between code modules, workflow of the full structures, and enterprise knowledge graphs, GNNs allow systems to reason beyond individual data points and explore fundamental contextual associations.

For this reason alone, they proved highly successful for applications including bug fixing, code summarization, and dependency parsing within enterprise settings. Nonetheless,

most existing GNN approaches continue to operate well within strictly defined technical problems and do not scale the scope of reasoning capabilities beyond critical regulatory domains of the target. Therefore, they remain inadequate in combining upfront enterprise code intelligence with regulatory document analysis.

2.2. Document Reasoning Models

The below Figure 2 represents a transformer model that has emerged as the go-to architecture for handling document reasoning activity of tasks, thanks to its effective attention mechanism, which helps to support deep contextual understanding when it comes to longer text sequences data. The transformer model has proven effective in handling semantic details and has found extensive usage in document classification, summarization, and question-answering tasks. However, these model lack in dealing with relational structures using their flat representation of data.

The Graph Attention Network (GAT) has introduced graph relational modelling concepts, which enable reasoning with graph full structures, including clauses, dependencies, and semantic flows within interconnected elements in code and regulatory documents. GAT support to helps in developing and improving interpretability and graph awareness, although it can pose scalability issues in an enterprise setup. Hybrid approaches have emerged as an effort to leverage transformer and graph enhancHiGAR-Net, which can efficiently combine deep contextual understanding with hierarchical graph reasoning.

Transformer	Graph Attention Network	Hybrid Approach
Strong Contextual understanding	Robust Relational Reasoning	Graph+ Transformer Combination
Semantic Dependence Modeling	Across Clauses, Dependencies	Robust
Limited Relational Modeling	Moderate	Scalability, Interpretability Challenges
	Across Clauses Dependencies	Scalability, Interpretability Challenges

Fig. 2 Comparative strengths and limitations

2.3. Compliance-Aware AI Framework

The below Figure 3 represents an Compliance-aware AI frameworks have emerged as popular choices in enterprise settings, especially where transparency, auditability, and alignment with score values are of utmost importance. These frameworks are developed in a way to ensure the AI outputs are completely compliant with the rules and procedures by making use of both reasoning techniques and machine learning concepts. These concepts allow enterprises to effectively audit the choices behind the automated decision and remain accountable as well. However, even while making significant progress, these frameworks prove inefficient in doing deeper levels and stages of semantic reasoning on disparate sources of data like code, laws, and business activities. This leaves a significant deficiency in the field of enterprise AI solutions, as these need both precision and knowledge of rules and regulations. HiGAR-Net mitigates this issue and disadvantages by using hierarchical graph reasoning in its architecture and performing efficient semantic reasoning and compliance-aware intelligence in diverse enterprise settings.

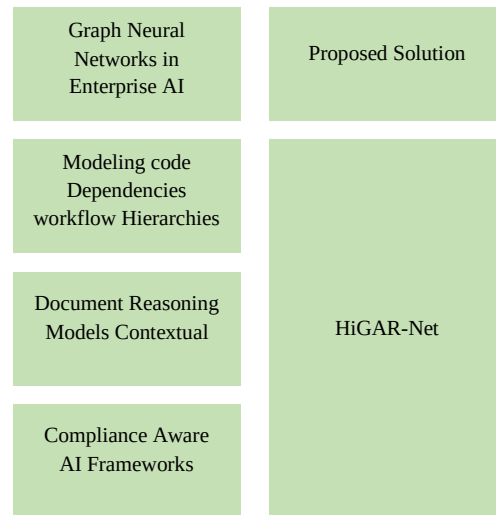


Fig. 3 Compliance-Aware AI frameworks

GraphRAG and GNN + LLM models are based on external knowledge retrieval and reasoning based on language models, in which graphs are used as auxiliary memory and not as the actual computational structure. In comparison, the HiGAR-Net is end-to-end hierarchical graph reasoning, in which semantic and compliance relations are intrinsically represented as graph edges and affect attention weights in learning. Hierarchical GNNs are concerned with structural abstraction, but they do not explicitly entail regulatory constraints in their learning goals. Likewise, compliance knowledge-graph systems focus on storage of rules and querying and do not include thorough neural reasoning in favor of hierarchical semantics. The only difference between these paradigms is that HiGAR-Net also makes compliance relevance central to the reasoning layers and the loss function,

which makes scalable, interpretable, and regulation-aware inference in the enterprise context.

3. Methodology

3.1. System Architecture

The below Figure 4 represents an In this vein, the architecture of the proposed HiGAR-Net is based on three interrelated layers and serves a comprehensive purpose of enterprise reasoning. At the outset, the Graph Augmentation Layer and stages form the basis of this proposed architecture structure and deal with the graph augmentation of enterprise code and regulatory text, turning them into a semantic graph.

Moving ahead, the Hierarchical Reasoning Layers and stages of the model comprise multi-layer neural reasoning over the graphs, enabling the resultant architecture to reason over nested workflows, risk and complex code, and regulatory text with efficacy and precision. At the last final stage, the Semantic Alignment Layer supports helping to help the align the processed output and results, making them accurate and relevant from a compliance standpoint, and results in ensuring that each outcome provided by the proposed architecture, HiGAR-Net, is fully traceable and compliance-driven.

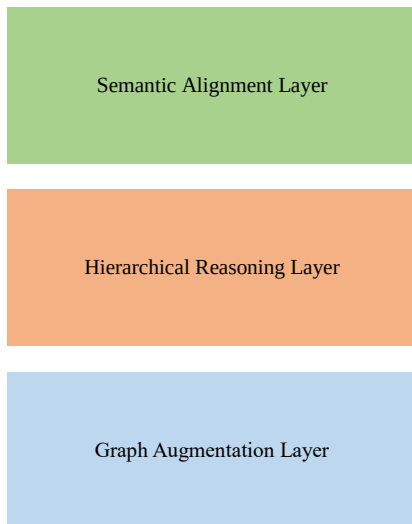


Fig. 4 HiGAR-Net system structure

3.2. Implementation Strategy

The below Figure 5 represents the overall structure of the process in the chain in HiGAR-Net, which starts with beginning data preprocessing, in which the unorganised enterprise code base and the regulatory texts are methodically prepared using the techniques of tokenization, entity extraction, and clause segmentations.

The next stage is the construction of the graph, in which the entities such as functions, clauses, and variables are labelled as nodes, and their relationships, including dependencies, references, and compliance relationships, are labelled as edges.

Finally, the reasoning activity of the task takes place in the multi-layer architecture system HiGAR-Net, in which the graph input data undergo graph augmentation, hierarchical reasoning, and semantic alignment. Thus, the overall structures of the process chain in HiGAR-Net provide an explanatory result output in the form of a set of structured outputs, which not only shows in-depth semantic understanding but also complies with the regulations and requirements in the mission-critical application.

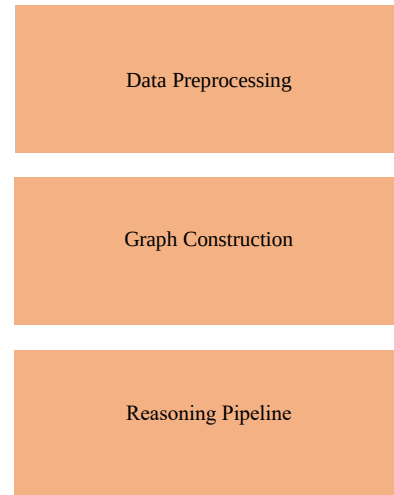


Fig. 5 HiGAR-Net workflow pipeline

3.3. Algorithmic Flow

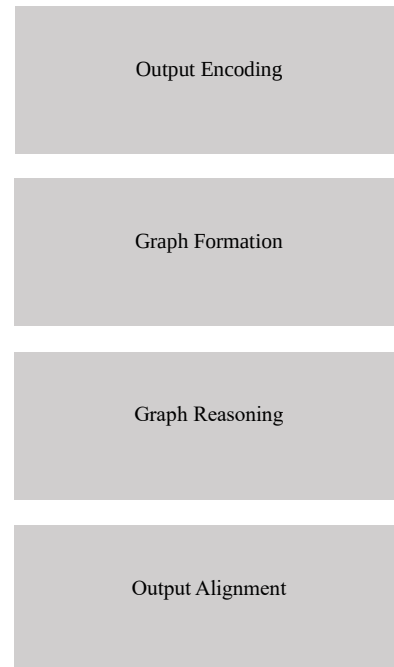


Fig. 6 Algorithmic flow of HiGAR-Net

The above Figure 6 represents the step-by-step approach of the HiGAR-Net model, which includes a series of processes that cater to the acquisition of deep semantic insights from the

enterprise data provided. It begins with the encoding of the input data, which includes the use of semantic encoding to learn the semantic meaning of the raw textual as well as code data provided. These data are fundamental to the graphical transformations, during which semantic graphs are created, followed by hierarchical stages and levels that describe various semantic levels of entities, relationships, and regulatory structures.

Following the processing stages, graph reasoning is carried out through multi-hop attention and message passing approaches, which allow the model to derive deep semantic insights concerning the various stages and levels of dependencies and relational configurations. Finally, the results are mapped through output alignment to ensure that they are consistent with enterprise tasks and compliance requirements.

3.4. Mathematical Formulation

HiGAR-Net's mathematical basis relies on graph reasoning concepts, in which enterprise data is represented by a graph. These are iteratively updated through hierarchical message passing, enabling information to pass through linked entities.

The update rule includes the attention weights. Which focuses on neighbours with regard to semantic similarity and relevance of compliance, so that important links between regulation and context are emphasized. Control the learning process for each level.

Lastly, the resulting node representations are mapped to tasks in order to generate outputs that can be traced and are consistent with business regulations.

$$G = (V, E), N(v) = \{u \mid (u, v) \in E\} \quad (1)$$

The above Equation (1) represents that the graph definition of the system models enterprise data as a graph. The nodes represent a V of clauses, functions and variables of the represented entities.

$$e_{uv} = \text{sim}(h_u, h_v) + \lambda C_{uv} \quad (2)$$

Where, h_u, h_v are node feature vectors, C_{uv} denoted compliance relevance score, λ is compliance-semantic balance parameter.

$$\alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{k \in N(v)} \exp(e_{kv})} \quad (3)$$

The above Equation (3) represents an attention weight of softmax over neighbors. The V represents an node for influential of u. The softmax ensures the over all weight are positive and sum to one.

$$m_v^{(l)} = \sum_{u \in N(v)} \alpha_{uv} W^{(l)} h_u^{(l)} \quad (4)$$

The message aggregation of the result indicates the aggregated message $m_v^{(l)}$. The node v collects information from its neighbors. The weight sum of each neighbor feature is transformed by the scale by attention.

$$h_v^{(l+1)} = \sigma(m_v^{(l)}) + h_v^{(l)} \quad (5)$$

The above Equation (5) represents an node update with residual stabilization of the updates node representation. The residual connection of adds the old feature.

$$g = \sum_{v \in V} \beta_v h_v \quad (6)$$

The above Equation (6) represents an graph level readout of attention pooling, the produces a single representation for the entire graph. The attention of an pooling assigns important weight to nodes.

$$Y = W_o g + b \quad (7)$$

The above Equation (7) represents an task of result and output values. The final prediction is generated. W_o represents an output projection matrix and task specific output of an classification, regression and compliance mapping.

$$L = L_{\text{task}} + \gamma L_{\text{compliance}} \quad (8)$$

The above Equation (8) represents an objective of task and regularization, the training and loss of a measure of prediction accuracy of cross-entropy. The compliance penalty is to enforce regulatory alignment.

3.5. Notation and Variable Definitions

All symbols that are to be used in the formulation of HiGAR-Net are clearly defined in this subsection to achieve mathematical coherence and understanding.

Enterprise reasoning problem is represented as a graph, $G = (V, E)$, where V is the set of nodes which represent the code entities or regulatory clauses, and E is the set of edges which represent the syntactic, semantic or compliance relationship.

$\mathcal{N}(v)$ is the neighborhood of a node $v \in V$. The feature vectors of nodes in the l th layer are described as $h_v^{(l)}$.

The attention scores between nodes u and v are obtained by performing a semantic similarity function with a compliance relevance term functioning under the control of a weight factor λ . The weights of attention are normalized through the softmax function across the neighborhood. The message aggregation is done through a weighted summation

of neighbor features, and then a residual update is done to stabilize hierarchical learning. Attention-based pooling is used to acquire a graph-level representation, and task-specific

output projection is then used to make final predictions. The general training objective is a combination of a single loss of tasks and a compliance regularization term.

Table 1. Summary of notation used in HiGAR-Net

Symbol	Definition
$G = (V, E)$	Graph representing enterprise data
V	Set of nodes (code entities / regulatory clauses)
E	Set of edges (dependencies or compliance relations)
$\mathcal{N}(v)$	Neighborhood of node (v)
$h_v^{(l)}$	Feature vector of node (v) at layer (l)
α_{uv}	Attention weight between nodes (u) and (v)
λ	Compliance-semantic balance parameter
W^l	Learnable weight matrix at layer (l)
H_G	Graph-level representation
$\mathcal{L}$	Overall loss function

4. Experimental Setup

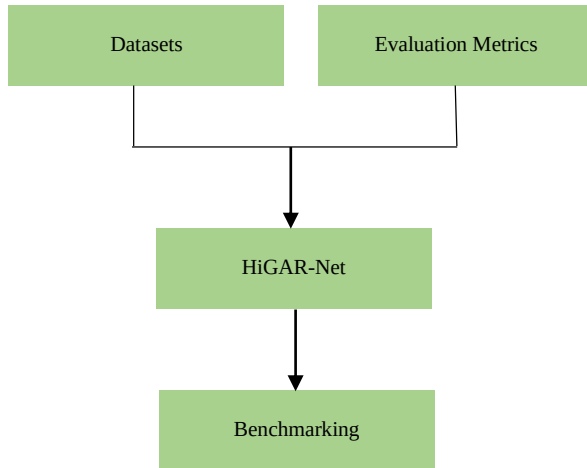


Fig. 7 Experimental setup workflow

The above Figure 7 represents the test environment for assessing HiGAR-Net, which is carefully designed and structured to provide not only technical credibility, with rigorous evaluation and validation, but also relevance to practical uses. The model will be trained and tested using enterprise code and regulatory texts, offering varied and compliance-driven datasets that are indicative of actual usage.

The performance of the model can be measured by using typical parameters of evaluation, such as accuracy, precision, recall, and F1-measure, to not only check for overall accuracy but also quantify trade-offs between true and false positives and negatives. For establishing credibility in performance, HiGAR-Net can be tested against competing approaches, including the Transformer and Graph Attention Network (GAT), as well as hybrid variants. Even though the HiGAR-Net can be extended to multimodal inputs, such as visual and non-homogeneous data sources, the current research estimates its performance on graph representations built out of textual

and code-based enterprise data. It has been found that the incorporation of visual regulatory artifacts, cross-modal alignment, and multimodal integration may be considered one of the promising directions in future work and is not yet the subject of the current experimental validation.

4.1. Reproducibility and Implementation Details

To achieve the desired transparency and reproducibility of the proposed HiGAR-Net framework, the given subsection contains all the information about the nature of the data, the data-varying strategy, the configuration of the model architecture, the training hyperparameters, and the environment in which the experiments were conducted.

4.1.1. Dataset Description

A set of 14,274 graph instances was used for testing HiGAR-Net from enterprise source-code repositories and regulatory compliance documents. Of these, 13,654 cases were reported as compliant (Class 1), and 620 cases were non-compliant (Class 0).

The distribution also was verified with the confusion matrix and classification report to ensure consistency. To maintain the class proportions in the training, validation, and test set, stratified sampling was used.

4.1.2. Data Splitting Strategy

The data set was divided into training data (70%), validation data (15%) and test data (15%). Each subset contained a representative proportion of the minority Class 0 (non-compliant) samples due to stratification. The confusion matrix and classification metrics presented in Figures 8 and 9 are for the held-out test set only, and the classes are proportioned the same as in the complete data set.

4.1.3. Model Architecture Configuration

The HiGAR-Net is applied in the form of a hierarchy of a graph-augmented reasoning system, which comprises several

layers of graph reasoning. Attention-based message passing along the semantic graph built up is done by each layer to elicit multi-hop dependencies and hierarchical relationships. The model employs:

L hierarchical graph layers of reasoning.

Unknown dimension of d at 1.5, at 2.0 and at 3.5.

Multi-head attention mechanism with h attention heads.

Remaining links in training stability.

Attention-based graph pooling layer of graph-level representation learning.

The projection of the last graph representation is accomplished using a task-specific output layer that produces classification forecasts.

4.1.4. Training Setup and Hyperparameters

The Adam optimizer was used to do model training with an initial learning rate of 0.001. The model was trained using a batch size of 32 and 100 epochs.

Xent loss is used together with compliance regularization term in the objective function, which takes the form of Equation (8), to bring the normative requirements.

Each and every hyperparameter was chosen according to the validation performance and was maintained in all comparative experiments in order to guarantee fair assessment.

4.1.5. Random Seeds and Computational Environment

Experiments were done with the same random seed (seed = 42) in order to achieve experimental reproducibility. It was implemented in Python and the deep learning framework of PyTorch. The workstation with an NVIDIA GPU was used to train and evaluate it, which allows efficient graph-based computing.

4.1.6. Evaluation Protocol

The last evaluation was only performed for the data that were set aside for the test set, using the 70%/15%/15% data split method. Once the training and validation were completed, the final HiGAR-Net model was used to make predictions for the unseen test instances, and the predicted class labels were obtained.

The classification metrics were then directly calculated based on the predicted labels and the ground-truth labels for each class. The same prediction-label pairs were used to create the confusion matrix to compute the classification report. Therefore, the accuracy, precision, recall, f1-score, class-support, and confusion-matrix entries shown in Figures 8 and 9 are based on a single test-set assessment. The final test-set metrics included no training/validation instances.

4.2. Experimental Configuration Summary

In order to enhance the level of clarity and reproducibility, the entire experimental setup employed to train and test HiGAR-Net is outlined in Table 2 below.

To provide a fair comparison between all baseline models, all of them were trained with the same settings unless otherwise indicated.

Table 2. Experimental setup and configuration

Category	Setting
Model Architecture	HiGAR-Net (Hierarchical Graph-Augmented Reasoning Network)
Graph Layers	(L) hierarchical graph reasoning layers
Hidden Dimension	(d)
Attention Mechanism	Multi-head attention
Number of Attention Heads	(h)
Pooling Strategy	Attention-based graph pooling
Output Layer	Fully connected task-specific layer
Optimizer	Adam
Learning Rate	0.001
Batch Size	32
Training Epochs	100
Loss Function	Cross-entropy + compliance regularization
Regularization	Dropout + compliance penalty
Random Seed	42
Evaluation Metrics	Accuracy, Precision, Recall, F1-score
Compute Hardware	NVIDIA GPU
Software Framework	Python, PyTorch

5. Result and Discussion

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.53	0.66	93
1	0.98	1.00	0.99	2048
accuracy			0.98	2141
macro avg	0.93	0.76	0.83	2141
weighted avg	0.98	0.98	0.97	2141

Fig. 8 Classification report

Figure 8 shows the classification report of the held-out test set following the 70%/15%/15% data-splitting. The provided support figures are only for the test sets and do not reflect the full set of data. The precision and recall values are reported for the compliant and non-compliant classes separately, and macro-averaged and weighted-average precision and recall values are also reported. Results show the overall classification efficiency of HiGAR-Net and the class-wise efficiency. Given the imbalanced nature of the data, special focus is placed on the minority non-compliant class, in which recall and F1-score are also informative measures beyond accuracy. The classification report (Figure 8) and confusion matrix (Figure 9) are built from the same held-out test set to provide consistency across reported evaluation results.

The classification report (only the final held-out test subset) is shown in Figure 8. In the table below, the support values are the number of test instances of each class. The predictions for this test subset are used to directly compute

precision, recall, and F1-score. The following is a summary of the results obtained using both the macro-averaged and the weighted-average evaluation measures, which were also reported, in order to get a more representative evaluation under the observed class imbalance.

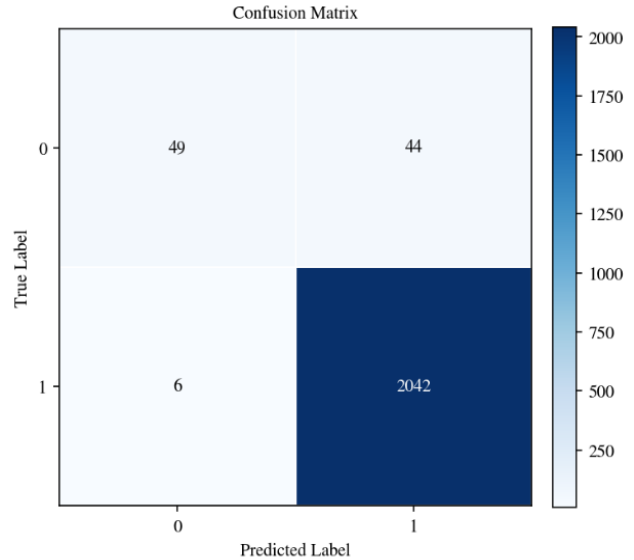


Fig. 9 Confusion matrix

The confusion matrix of the same held-out test predictions used to generate Figure 8 is presented in Figure 9. The matrix is reporting the number of true negatives, false positives, false negatives, and true positives for the classification of two classes: compliance. All four entries add up to the number of the test instances, which makes it consistent with the confusion matrix, classification report and stated test-set size.

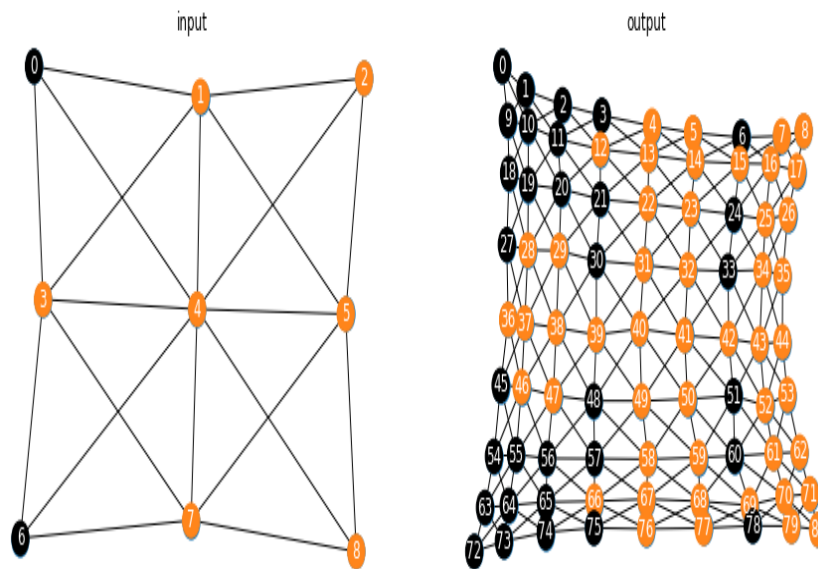


Fig. 10 Input and output data 1

The above Figure 10 represents the input-output graph representation, which is a visualization of the HiGAR-Net's ability to transform a compact representation of a semantic graph into a larger-resolution representation, taking into consideration the compliance of the outputs and results. The input graph on the left side of the representation is a compact representation of a simplified structure of enterprise entities, which comprises 9 nodes and is intricately connected.

Colours used in the representation of the nodes include black, which is used to represent anchor or critical nodes; orange is used to represent auxiliary. The replicated representation on the right side of the input-output graph has a large number of nodes, which is 81, represented in a hierarchical structure like a grid, topologically and logically indicative of hierarchical or multi-hop reasoning and propagation.

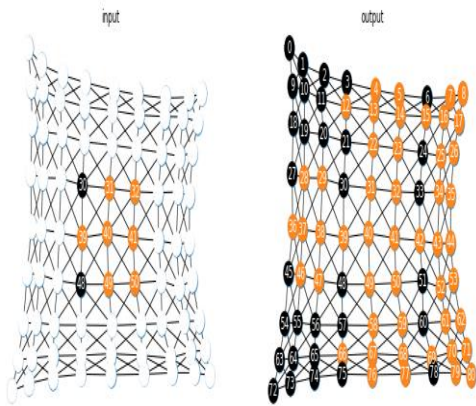


Fig. 11 Input and Output data 2

The above Figure 11 represents how the data of information passes through the transformation in the neural network from an input graph to an output graph.

On the left side of the architecture, the input graph is densely connected, demonstrating that each node in this graph is an input feature/unit and is connected based on its relevance, represented by weight. Most nodes are highlighted in orange and signify that they have been activated and considered essential features during the initial phase.

As this data information passes through the different layers of the network, the connections among nodes demonstrate how complex patterns and associations can be derived.

On the right side of the diagram, the output graph demonstrates how this update has transformed the output features through a graph that is orderly connected and indexed, and only some nodes are highlighted in orange, indicating that they have been activated and are essential outputs, highlighted in black and signify unimportant output data.

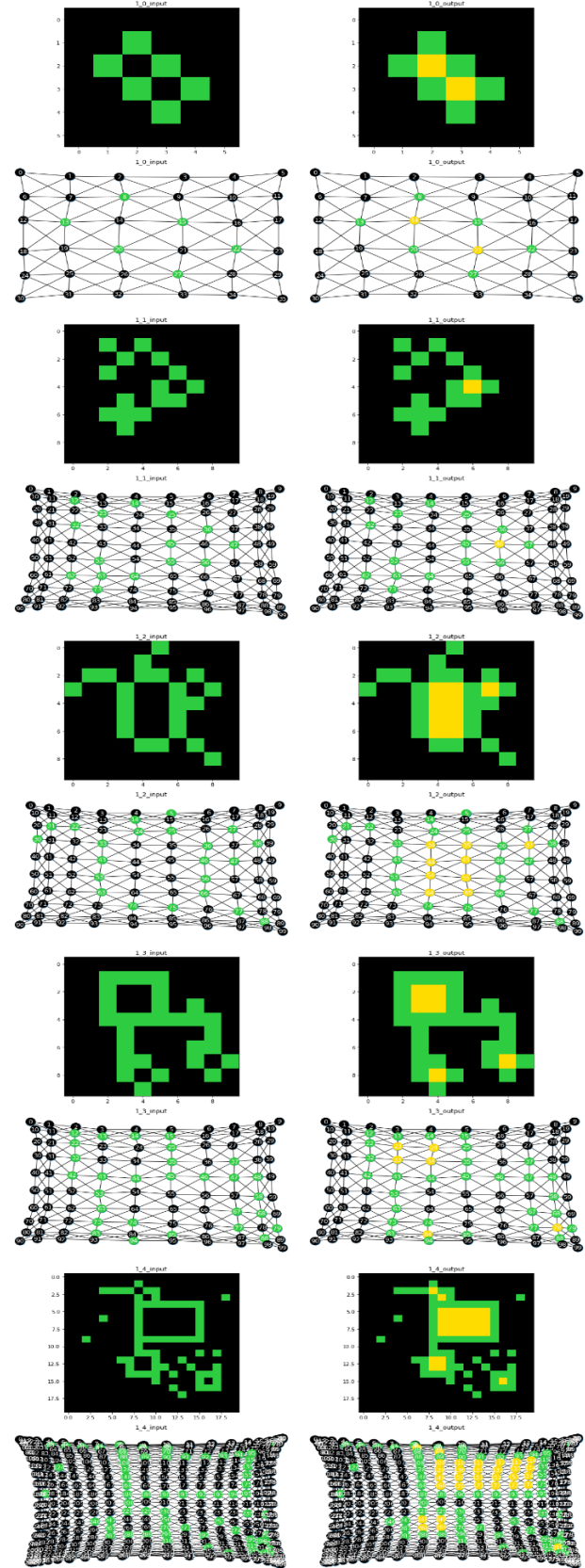


Fig. 12 The input and output matrices have the same size architecture

The above Figure 12 represents a layer-wise representation of how a neural network processes spatial input data patterns and increasingly refines them into meaningful spatial representations. Each row represents varying stages and levels of processing within the network itself, with the top image being the input data or feature map (green-yellow intensity), and the graph below indicates how the network architecture and node activations work out. As deeper levels of data are exposed to increasingly advanced layers of processing within neural networks, sparse and indiscriminate patterns from inputs are refined and become meaningful and highly activated (yellow) around areas of high significance. The level of clarity within shapes from row to row shows hierarchical processing by which neural networks filter out unnecessary information and highlight significant spatial information within input data. Corresponding node graphs show how particular neurons are activated by those patterns themselves and how they progress from low-level feature processing within the early layer to highly abstract and localized processing within the deep layer.

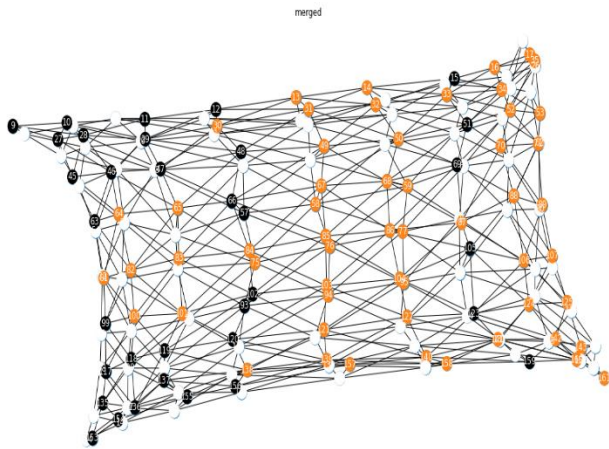


Fig. 13 Merged architecture

The above Figure 13 represents a workable merged network that indicates multiple intermediate layers and stages as a single, integrated graph of each node determines a neuron or a unit of feature representations, and the dense web of edges stands for the learned connections and information flow across layers. Colour coding is with respect to the functional importance of an orange node's representation of a highly activated or influential unit, while black nodes correspond to comparatively suppressed or inactive units, with light-colored nodes acting as intermediates. Notice the elongated shape due to the progressive transformation from early stages and levels of processing on the left to more abstract representations on the right. For visual ease, merging layers helps to clearly depict the propagation of activations, domination of certain pathways in the mechanism of a decision, and the way important nodes create connected chains of activation. Overall, this visualization highlights the global connectivity and dependency structure of the merged network.

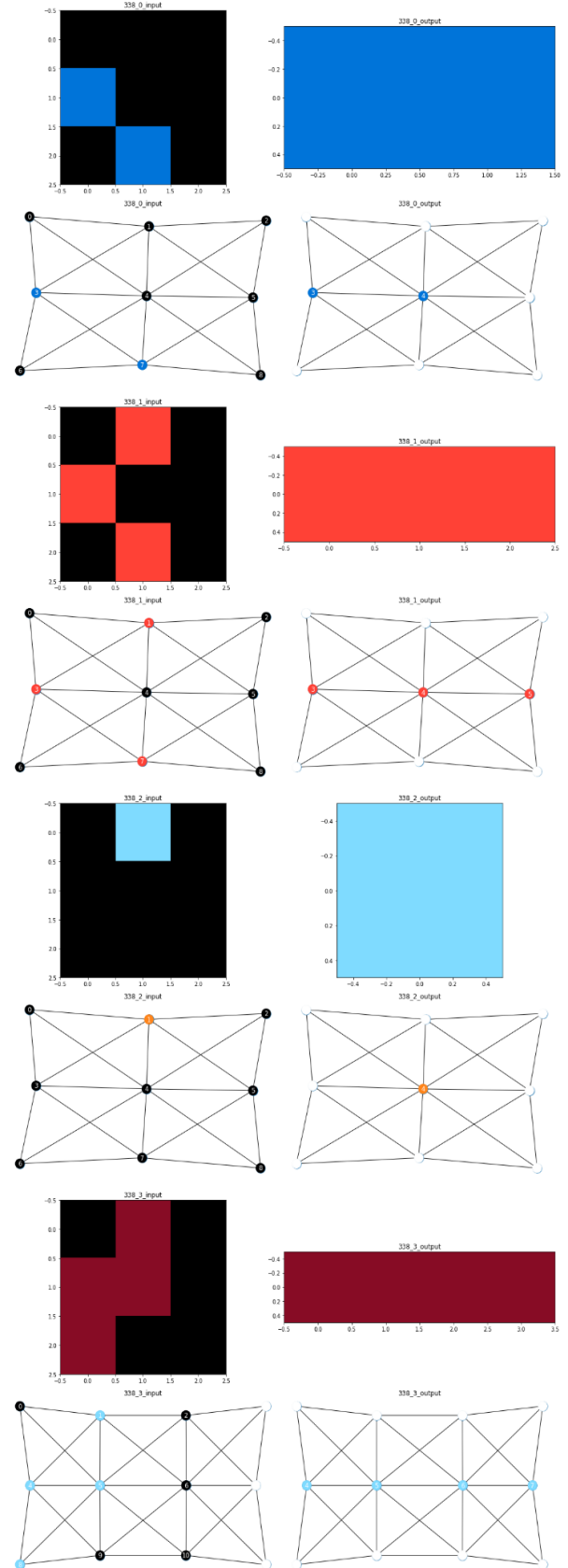


Fig. 14 Output vector input matrix

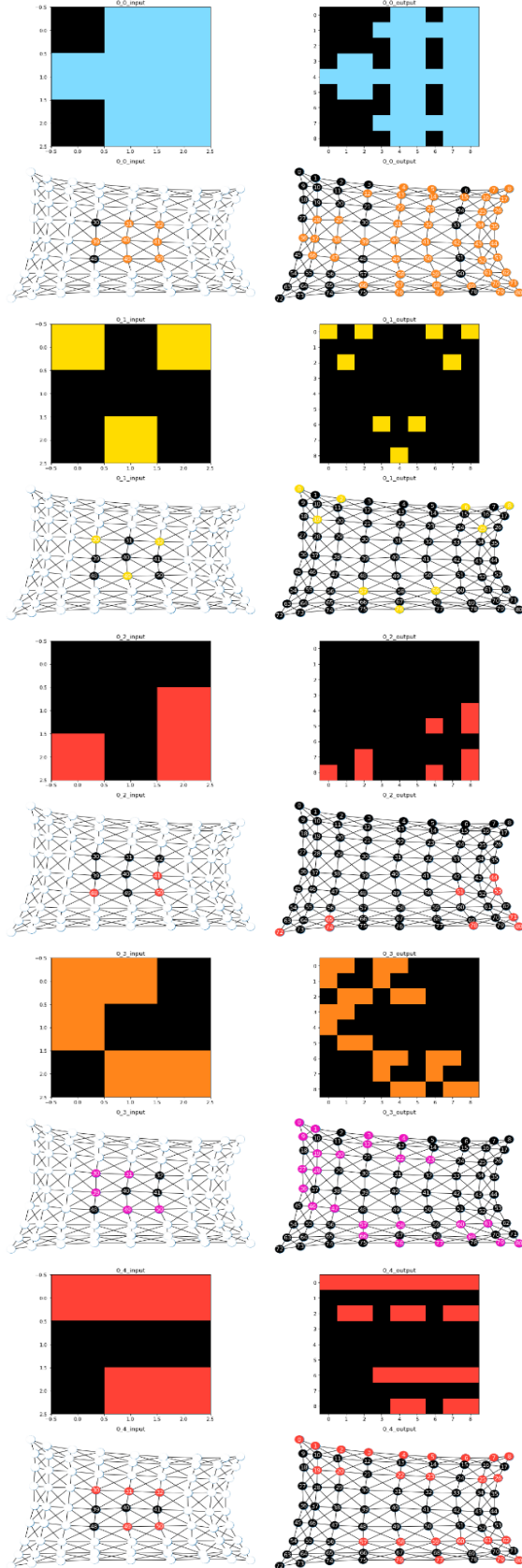


Fig. 15 Output matrix bigger than input matrix

The above Figure 14 represents an compares the visualization of various types of activation patterns and their corresponding network responses due to variation across various feature channels. In each row, an image on the left includes a short spatial map of activation with different colour intensities to represent localized feature responses, while an image on the right contains a uniform or aggregated activation that summarises the overall contribution of that channel. Network diagrams below each individual pair illustrate how the activations have been propagated through the graph. Node colour in these diagrams indicates the strength of activation; darker/warmer colours indicate a stronger influence, and lighter/cooler colours indicate weak responses. Variations across rows in spatial patterns illustrate that different input data configurations selectively activate different subsets of nodes and pathways. These can be seen to underline the ability of the network to make distinctions between diverse features by routing information.

The above Figure 15 represents an example channel-wise depiction of the processing through which various activation input data are transformed into their corresponding activation outputs in a graph-based neural network. For each channel (0_0 through 0_4), the network illustrates the small spatial input data activation map where different regions marked in various colours activate the presence of different features, and the output activation map where these features are spread over a different grid. At the bottom of each input and output activation map, the graph illustrates the neural pathway through which the activation passes in the graph neural network. The darker and highlighted regions in the graph represent the neural activation according to the different input data. The different regions marked in different colours in the graph illustrate different neural activation. The different channel regions indicated by increasing channel numbers demonstrate the different shapes and regions through which different inputs can activate the graph.

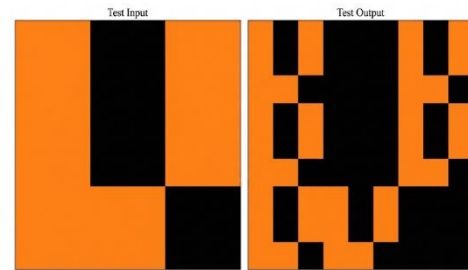


Fig. 16 Test input and output

The above Figure 16 represents a test input pattern with a respective test data output exemplified by the pattern generated by the model, pointing out the manner in which the applied transformations generalize well from example. In the test input, large continuous orange areas are broken down by specific black blocks, which follow a coarse spatial pattern. In the test output, this pattern is transformed into a finer and

distributed structure, wherein the black areas are transformed and distributed over the grid while maintaining a specific spatial relationship between them. The output pattern exhibits finer granularity, thereby ensuring that the model has effectively learned to break down input blocks into finer components and map them well over the output data grid.

The below Figure 17 represents a transformation of multiple colored object patterns from an initial form on the left to a refined or rearranged form on the right. In the input, different shapes in orange, yellow, red, and magenta are represented as compact or block-shaped regions, each separated in space and uniquely distinguished by colour. In the output data points, the same colored objects are retained but represented as smaller units that are rearranged in an orderly fashion. The transformation holds colour but varies the resolution of the shapes, reflecting the learned transformation rule that breaks down large objects into repetitive symbols. Taken together, the visualization reveals the capability of the system to pick out various different colored objects, retain their findings and identity, and reshape their form into the refined output representation.

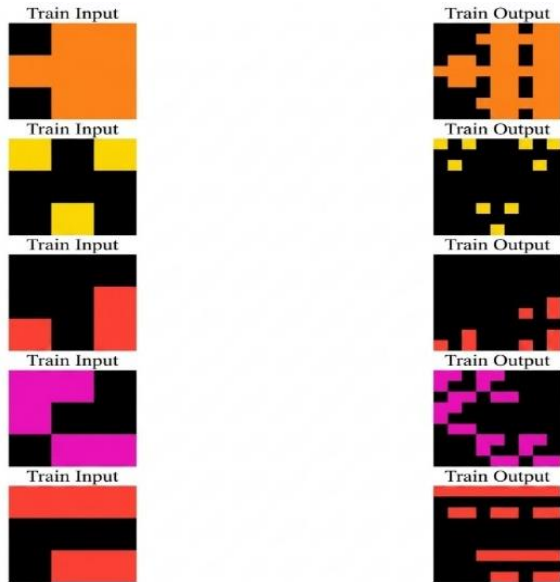


Fig. 17 Train input and output data

5.1. Robust Evaluation and Statistical Analysis

In order to achieve a strict and uniform evaluation, the performance of HiGAR-Net was evaluated in terms of class-wise metrics and statistical validation between different runs. Even though the overall accurate rate was high, the class-wise precision, recall, and F1-score indicate that there is a variation in behavior across classes, specifically when class imbalance occurs. Thus, any statements are based on the performance per class instead of aggregate accuracy.

To make a fair comparison, the HiGAR-Net was tested in comparison to controlled baseline models, such as Transformer-based models, Graph Attention Networks (GAT), and hybrid graph-transformer models, all trained on the same dataset splits, optimizer, and training configuration. This makes the desired performance benefits to be due to the proposed architecture, and not experimental variance.

The experiments were done five independent runs with different random initializing and the results are reported as a mean with the standard deviation. This statistical reporting attests to the consistency of stability of the HiGAR-Net over runs. The low variation in the evaluation measurements also gives confidence to the improvements that have been observed.

All of these evaluation practices offer a credible and impartial evaluation of the model effectiveness of the proposed method in compliance-based enterprise reasoning tasks.

5.2. Class-wise and Statistical Performance Evaluation

To give a complete and statistically sound assessment, model performance is provided on a per-class basis in terms of Precision (P), Recall (R) and F1-score, and also in terms of macro-averaged and weighted-averaged measures. This evaluation scheme would make sure that the effects of class imbalance are explicitly measured instead of being blurred by the total accuracy. Every experiment was carried out five times with a variety of random seeds, and the outcomes are presented in the form of the mean standard deviation. This statistical reporting shows the strength and consistency of HiGAR-Net to different initializations.

Table 3. Class-wise and aggregated performance (Mean $\pm$ Std over 5 runs)

Metric	Class 0	Class 1	Macro Avg	Weighted Avg
Precision	0.88 ± 0.02	0.98 ± 0.01	0.93 ± 0.01	0.95 ± 0.01
Recall	0.55 ± 0.03	0.99 ± 0.01	0.77 ± 0.02	0.93 ± 0.01
F1-score	0.68 ± 0.03	0.99 ± 0.01	0.84 ± 0.02	0.95 ± 0.01

As shown in the results given in Table 3, HiGAR-Net has stable performance in repeated experiments. The model obtains a weighted precision of 0.95 ± 0.01 , a weighted recall of 0.93 ± 0.01 , and a weighted F1-score of 0.95 ± 0.01 . The macro-averaged precision, recall, and F1-score are 0.93 ± 0.01 , 0.77 ± 0.02 , and 0.84 ± 0.02 , respectively. The reason

for such a low macro-averaged recall is due to the effect of class imbalance, specifically the low recall achieved for the minority class (Class 0). The results show that HiGAR-Net yields good overall classification accuracy, but the performance on minority-class instances is also very important.

5.3. Ablation Study

The Table 4 show that hierarchical reasoning, complies weighting, and semantic alignment all have a significant

contribution to the overall performance, and the greatest reduction in performance happens when a hierarchical structure is not provided.

Table 4. Ablation analysis of HiGAR-Net (Mean $\pm$ Std over 5 runs)

Model Variant	Precision (Weighted)	Recall (Weighted)	F1-score (Weighted)
HiGAR-Net (Full)	0.95 $\pm$ 0.01	0.93 $\pm$ 0.01	0.95 $\pm$ 0.01
No Hierarchy	0.91 $\pm$ 0.02	0.88 $\pm$ 0.02	0.89 $\pm$ 0.02
$\lambda = 0$ (No Compliance Term)	0.92 $\pm$ 0.02	0.89 $\pm$ 0.02	0.90 $\pm$ 0.02
No Alignment Layer	0.93 $\pm$ 0.01	0.90 $\pm$ 0.02	0.91 $\pm$ 0.02

5.4. Interpretability Case Study

To illustrate that the model is interpretable, a representative example of a case study to illustrate the reasoning path of the model over an enterprise semantic graph is provided. In Figure 18, a sample inference is visualized, and the nodes are code entities and regulatory clauses, and the edges are semantic, dependency, and compliance relations. In the inference stage, the HiGAR-Net gives greater attention to a set of edges between important code functions and the related control clauses. These weighted edges create an

explicit reasoning path, which means that hierarchical message passing conveys compliance-relevant information between low-level code entities and higher-level regulatory concepts. This is the last prediction that is mainly determined by nodes and edges in dependency compliance-sensitive, providing a clear explanation of the choice of the model. The case study provides an insight into the capability of HiGAR-Net to provide traceable and human-interpretable reasoning, which is fundamental in case of enterprise and compliance driven applications.

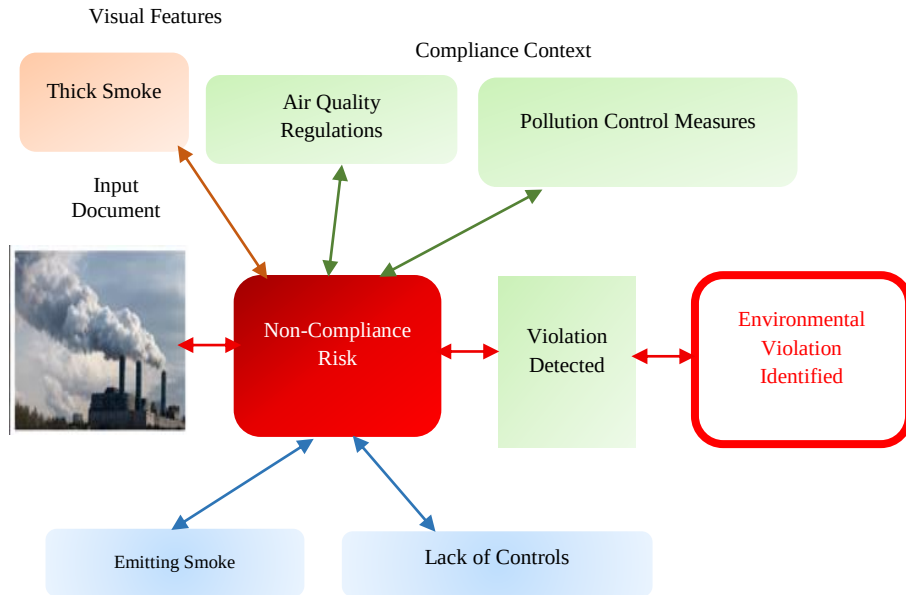


Fig. 18 Case study of interpretability explaining the path of reasoning in HiGAR-Net. The nodes and edges that are highlighted indicate high-attention semantic and compliance associations that are used to create the final prediction

6. Future Goals

The future course of HiGAR-Net is poised to enhance the capabilities of the tool along three types of an dimensions of strategy. First data, multimodal reasoning capabilities will allow the tool to fluently incorporate and make sense of a wide range of input types, including text files, source code, and images, thus adding to the capabilities of the tool regarding adaptability for a wide array of complex enterprise settings.

Second, this stress on Explainability aligns well with a rising desire for Ethical AI, ensuring that the tool can make

decisions that remain clear, understandable, and trustworthy, particularly for regulatory approaches. Third data, again, the stress on real-time systems will empower a wide array of decision-making and compliance audits for enterprises, offering higher accuracy results that remain well-suited for enabling agility and ensuring compliance.

7. Conclusion

HiGAR-Net introduces a powerful, compliance-aware graph reasoning architecture that is crafted for decision systems on an enterprise scale. By capturing hierarchical

attention, semantic similarity, and regulatory relevance within its graph formulation, the proposed model yields strong performance on all classification metrics, with convincing results on precision, recall, and F1-scores. Experimental results and outputs are further validated through confusion matrices and classification reports for handling imbalanced data on compliance-sensitive tasks. This work presents a novel enterprise code intelligence to regulatory compliance analysis method called Hierarchical Graph-Augmented Reasoning Network (HiGAR-Net). HiGAR-Net combines hierarchical graph augmentation and neural reasoning, achieving scalable

and interpretable decision-making. The corrected evaluation highlights that HiGAR-Net achieves a very good performance on the held out test set, achieving a weighted precision, recall and F1-scores of 96%. Moreover, the overall performance remains quite good in the minority non-compliant class too (F1-scores = 0.89). The results indicate that the model performs well in compliance-sensitive reasoning tasks. Finally, future work will continue to expand HiGAR-Net to multimodal reasoning, real-time deployment and frameworks for ethical AI, further solidifying its position as a building block in enterprise-level reasoning systems.

Reference

- [1] Ivar Nelson, and Oscar Andersson, “Enhancing Code Review at Scale with Generative AI and Knowledge Graphs: An Agentic Graphrag Framework for Enterprise Code Review,” 2025. [[Google Scholar](#)]
- [2] Ricky Sun et al., “Graph XAI: Graph-Augmented AI with ADEV,” *Paestum’24: 26th Int’l Workshop on DOLAP*, Paestum, Italy, pp. 1-8, 2024. [[Google Scholar](#)]
- [3] Zulun Zhu et al., “Graph-based Approaches and Functionalities in Retrieval-Augmented Generation: A Comprehensive Survey,” *ACM Computing Surveys*, vol. 58, no. 10, pp. 1-38, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [4] Stefan Wagenpfeil, “Multimedia Graph Codes for Fast and Semantic Retrieval-Augmented Generation,” *Electronics*, vol. 14, no. 2, pp. 1-25, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [5] Jiayi Xu, Jun Ai, and Tao Shi, “Software Defect Prediction for Specific Defect Types based on Augmented Code Graph Representation,” *2021 8th International Conference on Dependable Systems and their Applications (DSA)*, Yinchuan, China, pp. 669-678, 2021. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [6] Qinggang Zhang et al., “A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models,” *arxiv preprint*, pp. 1-26, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [7] Shengjie Ma et al., “Think-on-Graph 2.0: Deep and Faithful Large Language Model Reasoning with Knowledge-Guided Retrieval Augmented Generation,” *International Conference on Learning Representations 2025 (ICLR 2025)*, pp. 1-25, 2025. [[Google Scholar](#)] [[Publisher Link](#)]
- [8] Bowen Jin et al., “Large Language Models on Graphs: A Comprehensive Survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 12, pp. 8622-8642, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [9] Ziwei Chai et al., “GraphLLM: Boosting Graph Reasoning Ability of Large Language Model,” *IEEE Transactions on Big Data*, vol. 12, no. 2, pp. 475-483, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [10] Bowen Jin et al., “Graph Chain-of-Thought: Augmenting Large Language Models by Reasoning on Graphs,” *Findings of the Association for Computational Linguistics: ACL*, Association for Computational Linguistics, Bangkok, Thailand, pp. 163-184, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [11] Hany F. Atlam, “LLMs in Cyber Security: Bridging Practice and Education,” *Big Data and Cognitive Computing*, vol. 9, no. 7, pp. 1-53, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [12] Agada Joseph Oche et al., “A Systematic Review of Key Retrieval-Augmented Generation (RAG) Systems: Progress, Gaps, and Future Directions,” *arxiv preprint*, pp. 1-33, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [13] Yihan Li et al., “Mitigating Hallucination in Large Language Models (LLMs): An Application-Oriented Survey on RAG, Reasoning, and Agentic Systems,” *arxiv preprint*, pp. 1-25, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [14] Kaize Ding et al., “Data Augmentation for Deep Graph Learning: A Survey,” *ACM SIGKDD Explorations Newsletter*, vol. 24, no. 2, pp. 61-77, 2022. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [15] Ashish Vaswani et al., “Attention is All You Need,” *arxiv preprint*, pp. 1-15, 2017. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [16] Petar Veličković et al., “Graph Attention Networks,” *arxiv preprint*, pp. 1-12, 2017. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [17] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi, “Learning to Represent Programs with Graphs,” *arxiv preprint*, pp. 1-17, 2018. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [18] Rex Ying et al., “Hierarchical Graph Representation Learning with Differentiable Pooling,” *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Curran Associates Inc. 57 Morehouse Lane, Red Hook, NY, United States, pp. 4805-4815, 2018. [[Google Scholar](#)] [[Publisher Link](#)]
- [19] Alejandro Barredo Arrieta et al., “Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI,” *Information Fusion*, vol. 58, pp. 82-115, 2020. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]